INT J COMPUT COMMUN, ISSN 1841-9836
Vol.7 (2012), No. 3 (September), pp. 494-508

Mining Temporal Sequential Patterns Based on Multi-granularities

N. Li, X. Yao, D. Tian

Naiqian Li, Xinhui Yao, Dongpin Tian
Department of Computer Science
Baoji University of Arts and Sciences
Baoji 721016, Shaanxi, China
E-mail: xalnq@hotmail.com, baojiyxh@163.com,
tdp211@163.com

Abstract:
Sequential pattern mining is an important data mining problem that can extract frequent subsequences from sequences. However, the times between successive items in a sequence is typically used as user-specified constraints to pre-process the input data or to prune the pattern search space. In either cases, the times cannot be used to identify item intervals of sequential patterns. In this paper, we introduce a form of multi-granularity sequence patterns, which is a sequential pattern where each transition time is annotated with multi-granularity boundary interval and average time derived from the source data rather than the user-predetermined time interval or only a typical time. Then we present a novel algorithm, MG-PrefixSpan, of multiple granularity sequential patterns based on PrefixSpan[, which discovers all such patterns. Empirical evaluation shows that MG-PrefixSpan scales up linearly as the size of database, and has a good scalability with respect to the length of sequence and the size of transaction.
Keywords: Data Mining Algorithm, Sequential Pattern Mining, Sequential Data, Time Granularity, Temporal Patterns.

1 Introduction

Among various types of data mining applications [1,2], the sequential pattern mining, which discoveries interesting sequential patterns hidden in sequence of events, is an important data mining problem with broad applications, including market analysis, decision support, the prediction of occurrences of recurrent illnesses, system performance analysis and telecommunication network analysis etc.

The problem of mining sequential patterns was first proposed by Agrawal and Strikant [3]: Given a data set of sequences, each sequence is a list of transactions, where each transaction is a set of items. The sequential pattern mining is to find all subsequences that is more frequent than a user-specified minimum support threshold while maintaining their item occurrence order.

For example, in the database of a book-club, a sequential pattern might be “5 percent of customers bought ‘Foundation’, then ‘Foundation and Empire’, and then ‘Second Foundation’” [4]. Although the discovered sequential patterns reveal what items are frequently bought together and in what order, they cannot reveal how long time the items will be bought after the preceding items. Unfortunately, not knowing the time means that we cannot exactly predict when the next purchase will happen. In addition, some sequential patterns could occur in different periods with different time granularities. For example, “HP stock could rise within 5 days after IBM stock rose" and “HP stock could fall within 6 month after IBM stock rose", which reveal HP stock rise or fall with respect to IBM stock rise at different time granularities (day and month), are two useful different patterns. However, these traditional sequence patterns can only tell us “HP stock could rise after IBM stock rose" and “HP stock could fall after IBM stock rose". This situation means that the two patterns are useless. Another situation may be “HP stock

could rise 5 days later after IBM stock rose" and “HP stock could rise 6 months later after IBM stock rose". Although these two patterns are completely different with regard to different time granularities, these traditional sequence patterns could treat the two patterns as the same pattern “HP stock could rise after IBM stock rose", which make the extracted pattern less precise and some useful information lost.

Given the above reasons, in this paper we generalize the problem definition given in [1,8,9,13] to incorporate the maximum, minimum and average time between successive transactions, which are derived from the source data, and different time granularities in traditional sequential patterns. We present a novel algorithm of multiple granularity sequence patterns based on PrefixSpan [6] , called MG-PrefixSpan, which discovers all such patterns. Empirical evaluation shows that MG-PrefixSpan scales up linearly as the size of database, the length of sequence and the size of transaction.

The rest of this paper is organized as follows. Related work is discussed in section 2. We give a formal description of the problem of mining temporal sequence patterns based multiple granularities in section 3. In section 4, we describe MG-PrefixSpan, an algorithm for finding such patterns, and then Section 5 provides empirical evaluation of the performance of MG-PrefixSpan. Finally, we conclude the paper in section 6.

2 Related Work

Sequential pattern mining, in general, can be grouped into two categories. One category, called un-temporal sequence pattern mining or traditional sequence pattern mining, considers only the item occurrence order in a sequential pattern, but does not deal with time-related data [1, 3–6]. The other category, called temporal sequence pattern mining, consider not only the item occurrence order in a sequence pattern, but also the time between successive items in a sequential pattern, such as [8,9,11–13].

2.1 Un-temporal Sequence Pattern Mining

Agrawal and Strikant [3] introduced the notion of sequential pattern mining, and based on the property that any sub-pattern of a frequent pattern must be frequent, three Apriori-based algorithms were proposed: AprioriSome, DynamicSome and AprioriAll. Two of these algorithms were designed only to find maximal sequential patterns. The third algorithm, AprioriAll, finds all patterns. Briefly, AprioriAll is decomposed into two phases: (1) generating candidate sequences; (2) scanning the sequential database to check the support of each candidate to determine frequent sequence patterns according to minimal support threshold. Although AprioriAll is not efficient, it is the basis of many efficient algorithms developed later. SPADS [5] is an algorithm proposed to find frequent patterns using efficient lattice search technology and simple joins. It decomposes the original search space (lattice) into smaller pieces (sub-lattices), which can be processed independently in the main memory. Due to adopting a vertical id-list database format to count the number of frequent patterns, all the sequential patterns are discovered with only a few passes over the database. SPADS outperforms AprioriAll [2,3]. PrefixSpan[6] is another more efficient algorithm for mining sequential patterns comparing with the aprior-based algorithm AprioriAll and SPADS, especially in dealing with very large databases. It mainly adopts a projection-based, sequential pattern-growth method to make the database for next pass much smaller and consequently make the algorithm more speedy. Also in PrefixSpan there is no need to generate candidates, only recursive projection of database according to their prefix. Our method is based on the PrefixSpan algorithm.

Some have tried to exert constraints on the mining of sequential patterns so that only those sequential patterns interesting to users are discovered rather than the whole possible sequential patterns[2,5,6,7]. Strikant and Agrawal [4] generalized their definition of sequential patterns in [3] to integrate with time constraints, sliding time window, and user-defined taxonomy, and proposed algorithm GSP. Mannila et al. [7] proposed a method of mining frequent episodes in a sequence of events, in which episodes are essentially constraints on events in form of acyclic graphs. Garofalakis et al. [8] proposed a family of

SPIRIT algorithms of mining user-specified sequential patterns by using regular expression constraints. Pei et al. [9] developed an extended framework based on a sequential pattern growth for constraint-based sequential pattern mining. Although time between successive items is typically used as a user-specified constraint to shrink the pattern search space to make the computation more efficient, it is not used in the output frequent sequence patterns.

2.2 Temporal Sequence Pattern Mining

Yoshida et al.[10] proposed a notation of delta pattern, which is a temporal sequence pattern with temporal constraints in the form $A \overset{[0,7]}{\rightarrow} B \overset{[3,5]}{\rightarrow} C$ of bounding intervals. An example of delta pattern has the form denoting a sequential pattern $A \rightarrow B \rightarrow C$ that frequently appears in the database with transition times from A to B and from B to C that contained in [0, 7] and in [3, 5] respectively. However, Yoshida et al. only provided a heuristics for finding some frequent delta patterns, and did not investigate the problem of finding all of them. Along the same direction, Chen et al. [11] introduced a form of temporal sequence pattern by inserting pseudo items into the original sequential pattern. Pseudo items are user-defined time interval segmentations in advance. When counting the support of a sequential pattern, only the sequence, in which the pseudo item between successive items is same, supports the sequential pattern. Two algorithms, I-Apriori and I-PrefixSpan, were proposed. I-Apriori is based on Apriori algorithm [12], and I-PrefixSpan is based on PrefixSpan[6]. Hirate et al. [13] proposed generalized sequential pattern mining with item intervals. They extended sequences which are defined by inserting pseudo items based on the interval itemization function and exerting four interval constraints on items. However, as they adopted some user-predefined pseudo items or constraints on the time intervals between successive items, it is difficult for a user to specify optimal constraints related to item interval and cannot reveal the time interval between successive items in the patterns precisely, it may result in some useful sequential patterns not being found. Giannotti et al. [14] introduced another form of sequential pattern, called Temporally-Annotated Sequence, TAS in short, where each transition is annotated with temporal information representing a typical time derived from the source data. For instance, TAS $A \overset{t_1}{\rightarrow} B \overset{t_2}{\rightarrow} C$ denotes the fact that a sequential pattern $A \rightarrow B \rightarrow C$ frequently appears in the database and that the time for getting from event A to B is close to t_1 and from event B to C is close to t_2. Although this method using typical time to annotate transition between two successive events can reveal how much time the event will occur after the preceding event, they cannot distinguish completely patterns with regard to different time granularities. For example, from event A to B is close to t_1 days, other from event A to B may be close to t_2 months, these two different patterns are treated as the same. It is useful to be able to distinguish the patterns to understand not only what event will follow, but also when these events will occur.

Finally, we mention the work in [15], where event structures that have temporal constraints with multiple granularities are introduced. Event structures essentially are used as a flexible user-defined constraint specification to define the pattern discovery problem with these structures that enable users to focus on their interested sequential patterns. This method can only find the patterns that satisfy the event structures. Furthermore, an event structure consists of a number of variables representing events and temporal constraints among these variables, an efficient event structure is difficult to define beforehand even if you are a domain expert.

In our work, we introduce a new form of sequential pattern with multi-granularities, which is a sequential pattern where each transition is annotated with multi-granularity boundary interval and average time derived from the source data rather than user-predetermined time intervals [8,9] or only a typical time [14]. We also define the pattern discovery problem involving multi-granularities for these pattern and study efficient algorithm based on PrefixSpan to solve it.

3 Problem Formulation

3.1 Time granularity

In order to formally define temporal sequence pattern that involves time granularities, we first review the notion of a time granularity [15].

Definition 1. A granularity is a mapping μ from the set of the positive integers (the time ticks) $\mathcal{R}$ to $\mathcal{R}^2$ (the set of absolute time sets) such that for all positive integer i and j with $i \leq j$, the following two condition are satisfied:

(1) $\mu(i) \neq \emptyset \wedge \mu(j) \neq \emptyset$ implies that each number in $\mu(i)$ is less than all the numbers in $\mu(j)$, and

(2) $\mu(i) \neq \emptyset$ implies $\mu(j) \neq \emptyset$.

Each set $\mu(i)$, if non-empty, is called a granule of the μ. Property (1) says that granules do not overlap and the order on time ticks follow the order on the corresponding granules. Property (2) says that the subset of the time ticks corresponding to the granules forms a set of contiguous integers. The set $\mu(i)$ of reals is said to be the ith tick of μ, or tick i of μ . For example, hour, day, week, month, and year, satisfy the above definition. We can also define more complex granularities like business-week, weekend and so on.

When dealing with temporal types, it is needed to determine the tick (if any) of a temporal type μ that covers a given tick z of anther temporal type v. Formally, for each positive integer z and temporal types μ and v, if $\exists$ z' (necessarily unique) such that $v(z) \subseteq \mu(z')$ then $\lceil z \rceil_v^\mu = z\prime$, otherwise $\lceil z \rceil_v^\mu$ is undefined[15]. In this paper, all timestamps in a sequence are assumed to be in terms of a fixed granularity g_0, and abbreviate $\lceil z \rceil_v^\mu$ as $\lceil z \rceil^\mu$ if $v = g_0$.

Definition 2. A multi-granularity schema is a tuple of the form $G_m = (g_m, g_{m-1}, \cdots, g_2, g_1)$, where each g_i $(1 \leq i \leq m)$ is a granularity.

For simplicity, we require that in G_m and g_0, each unit of g_i is contained in a unit g_{i+1} $(0 \leq i \leq m-1)$.

3.2 Temporal sequence and multi-granularity sequence pattern

Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of items. An itemset s_i is a non-empty subset of items (without loss of generality, we assume that items of an itemset are sorted alphabetically) denoted as $i_1, \cdots, i_k$, where i_j $(1 \leq j \leq k)$ is an item. A traditional data sequence is an ordered list of itemsets, which is sorted by the order of priority of the transaction time and denoted as $(s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n)$, where s_i $(1 \leq i \leq n)$ is an itemset [1,3]. In practice, a data sequence of a customer is also formed as an ordered list of itemsets and time stamps [11], we call it a temporal sequence.

Definition 3. A temporal sequence s is represented as $\langle (s_1, t_1) \rightarrow (s_2, t_2) \rightarrow \cdots \rightarrow (s_n, t_n) \rangle$, where s_i is an itemset and t_i stands for the time at which s_i occurs, $t_i < t_j$ for $1 \leq i < j \leq n$.

When adding the itemset time information in the sequence, the time interval value between any two elements in the sequence can be computed as follows:

$$T_{ij} = T_j - T_i, \text{where } 1 \leq i < j \leq n$$

We now formally define multi-granularity schema, multi-granularity sequence patterns and its' support, then formalize our novel mining problem as the discovery of all frequent multi-granularity sequence patterns in the database D.

Definition 4. Given a multi-granularity schema $G_m = (g_m, g_{m-1}, \cdots, g_2, g_1)$, a multi-granularity sequence pattern is represented as follows:

$$\alpha = \alpha_1 \overset{[L_1, M_1, U_1]\mu_1}{\rightarrow} \alpha_2 \overset{[L_2, M_2, U_2]\mu_2}{\rightarrow} \cdots \overset{[L_{n-1}, M_{n-1}, U_{n-1}]\mu_{n-1}}{\rightarrow} \alpha_n$$

where (1) α_i $(1 \le i \le n)$ is an itemset; (2) $\mu_i \in \{g_j | j = 1, \cdots, m\}$ $(1 \le i \le n-1)$ a granularity; (3) L_i, M_i, and U_i $(1 \le i \le n-1)$ are respectively the lower bound, average time and upper bound of the time that α_{i+1} occurs after α_1, α_2, $\cdots$ and α_i with the granularity μ_i. $[L_i, M_i, U_i]\mu_i$ is called temporal annotation of α_{i+1}, and the tuple $([L_i, M_i, U_i]\mu_i, \alpha_{i+1})$ also called a temporal item $(1 \le i \le n-1)$.

Example $\alpha = \alpha_1 \overset{[1,25,5]day}{\rightarrow} \alpha_2 \overset{[2,33,4]week}{\rightarrow} \alpha_3$ is a multi-granularity sequential pattern, where $\alpha_1, \alpha_2, \alpha_3$ are itemsets or events, day and week are time granularities. The pattern indicates that α_2 occurs in 1 day to 5 days or average 2.5 days after α_1, then 2 weeks to 4 weeks or average 2.5 weeks later, α_3 occurs.

The total number of items in a multi-granularity sequence pattern is referred as the length of the pattern. A multi-granularity sequence pattern whose length is k is also represented as k-multi-granularity sequence pattern.

Definition 5. Given a temporal sequence $s = (s_1, t_1) \rightarrow (s_2, t_2) \rightarrow \cdots \rightarrow (s_n, t_n)$ and a multi-granularity schema $G_m = (g_m, g_{m-1}, \cdots, g_2, g_1)$. Let α be a multi-granularity sequence pattern $\alpha = \alpha_1 \overset{[L_1,M_1,U_1]\mu_1}{\rightarrow} \alpha_2 \overset{[L_2,M_2,U_2]\mu_2}{\rightarrow} \cdots \overset{[L_{k-1},M_{k-1},U_{k-1}]\mu_{k-1}}{\rightarrow} \alpha_k$, where $\mu_i \in g_j | j = 1, \cdots, m$ $(1 \le i \le k-1)$. s is said to support(or contain) α if and only if there exists integers $1 \le i_1 < i_2 < \cdots < i_k \le n$ such that

(1) $\forall j 1 \le j \le k$, $\alpha_i \subseteq s_{i_j}$

(2) $\forall j 1 \le j \le k-1$, if $\mu_j = g_h$ $(1 \le h \le m-1)$ then $g_h(1) \le T_{i_j i_{j+1}} < g_{h+1}(1)$, otherwise $T_{i_j i_{j+1}} \ge g_m(1)$.

The condition (2), in fact, is a constraint which divides the time between successive items into non-overlap partitions according to the sizes of different time granules. Without this constraint, the too long or too short time interval between successive items can be treated as the same in a pattern, and may result in some useful patterns not being found.

Definition 6. Let $\alpha = \alpha_1 \overset{[L_1,M_1,U_1]\mu_1}{\rightarrow} \alpha_2 \overset{[L_2,M_2,U_2]\mu_2}{\rightarrow} \cdots \overset{[L_{k-1},M_{k-1},U_{k-1}]\mu_{k-1}}{\rightarrow} \alpha_k$ be a multi-granularity sequential pattern, G_m be a multi-granularity schema, and S_l be a set of temporal sequence that support $\alpha^{l+1} = \alpha_1 \overset{[L_1,M_1,U_1]\mu_1}{\rightarrow} \alpha_2 \overset{[L_2,M_2,U_2]\mu_2}{\rightarrow} \cdots \overset{[L_l,M_l,U_l]\mu_l}{\rightarrow} \alpha_{l+1}$ $(1 \le l \le k-1)$ in database D. The annotation $[L_1, M_1, U_1]\mu_l$ of α_{l+1} is defined as follows:

$$\begin{aligned} U_l &= \max_{s \in S_l} \max_{i_1, \cdots, i_{l+1}} \lceil T_{i_l i_{l+1}} \rceil^{\mu_l} \\ L_l &= \min_{s \in S_l} \min_{i_1, \cdots, i_{l+1}} \lceil T_{i_l i_{l+1}} \rceil^{\mu_l} \\ M_l &= \frac{1}{|S_l|} \sum_{i_1, \cdots, i_{l+1}} \lceil T_{i_l i_{l+1}} \rceil^{\mu_l} \end{aligned}$$

where $i_1, i_2, \cdots, i_l, i_{l+1}$ are the indexes of the elements in $s \in S_l$ that match the elements of α^{l+1}, $\mu_l \in G_m$.

Definition 7. Let D be the database of temporal sequence, G_m be a multi-granularity schema. For a multi-granularity sequence pattern α, its support in database D is defined by:

$$Support_D(\alpha) = \frac{|\{s | s \in D \wedge s \ supports \ \alpha\}|}{|D|}$$

α is said to be frequent if its support is no less than the user-specified minimum support threshold min-sup i.e. $Support_D(\alpha) \ge min - sup$.

Given a temporal sequence database D, multi-granularity schema G_m and user-specified minimum support threshold min-sup, the task of mining multi-granularity sequence pattern is to find all frequent multi-granularity sequence patterns in the database D.

4 Algorithm for Mining Multi-Granularity Sequence Patterns

In this section, we propose a novel multi-granularity sequence pattern mining algorithm based on the well-known PrefixSpan[6], called as MG-PrefixSpan. The PrefixSpan algorithm solves the pattern mining problem by a divide-and-conquer, pattern-growth principle as follows: for each frequent item a, a projection of the initial sequence database D is created, denoted by $D|a$, i.e., which (1) contains only the suffix of sequences in D with respect to a; (2) contains only the frequent items; (3) in general is much smaller than D. Then, for each frequent item b in $D|a$, appends b to a to form a sequential pattern a' which starts with a. Finally a new, smaller projection $D|a'$ can be constructed recursively and used for finding longer patterns starting with a'.

However, although the PrefixSpan algorithm is very efficient for mining sequential pattern, it cannot be used straightforwardly to find multi-granularity sequence patterns. We extend the sequence database projection operation in PrefixSpan to be able to deal with the temporal relationship between successive items in the sequential pattern based on multi-granularities. Before introduction of the algorithm MG-PrefixSpan, we first extend definitions of prefix, suffix, projection and projected database based on PrefixSpan[6].

Definition 8. Given a temporal sequence $s = \langle (s_1, t_1) \rightarrow (s_2, t_2) \rightarrow \cdots \rightarrow (s_n, t_n) \rangle$ and a multi-granularity schema $G_m = (g_m, g_{m-1}, \cdots, g_2, g_1)$. A multi-granularity sequence pattern $\alpha = \alpha_1 \overset{[L_1,M_1,U_1]\mu_1}{\rightarrow} \alpha_2 \overset{[L_2,M_2,U_2]\mu_2}{\rightarrow} \cdots \overset{[L_{k-1},M_{k-1},U_{k-1}]\mu_{k-1}}{\rightarrow} \alpha_k$ $(k \leq n)$ is a multi-granularity sequence pattern prefix of s if and only if

(1) $\alpha_i = s_i$ for $1 \leq i \leq k-1$;

(2) $\alpha_k \subseteq s_k$ and all the items in $(s_k - \alpha_k)$ are alphabetically after those in α_k;

(3) $\forall i 1 \leq i \leq k-1$, if $\mu_i = g_h$ $(1 \leq h < m)$ then $g_h(L_i) \leq T_{ii+1} \leq g_h(M_i)$.

For example, let multi-granularity schema be G_2 = (hour, minute), g_0 = second. If there is a temporal sequence $\alpha = \langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$, $\beta_1 = (c)$ is a multi-granularity sequence pattern prefix of and so is $\beta_2 \overset{[1,2.4,4]minute}{\rightarrow} (ab)$.

Definition 9. Given a temporal sequence s and a multi-granularity sequence pattern α such that s supports α. A subsequence β of s is called a projection of s with respect to α if and only if

(1) β has a multi-granularity sequence pattern prefix α, and

(2) there exists no proper super-sequence β' of β such that β' is a subsequence of s and also has a multi-granularity sequence pattern prefix α.

If a temporal sequence $\alpha = \langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$ is projected with respect to $\beta_2 \overset{[1,2.4,4]minute}{\rightarrow} (ab)$, then two projection are $\langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$ and $\langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$. If α is projected with respect to $\beta_1 = (c)$, then three different projections are $\langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$, $\langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$ and $\langle (cf, 6400) \rangle$ respectively.

In the above, a temporal sequence α that is projected with respect to a prefix produces more than one projection. To differentiate these projections from the same temporal sequence, the tag [Sid, item, t] is attached to each projected sequences, where Sid is the identifier of the temporal sequence, item is last item in the multi-granularity sequence pattern and t is the time of element in temporal sequence that matches the last element of the multi-granularity sequence pattern.

Consequently, if sequence $\alpha = \langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$ is projected with respect to the β_2, then different projections are presented as follows:

[Sid, b, 80]: $\langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$;

[Sid, b, 320]: $\langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$.

Definition 10. Given a temporal sequence s and a multi-granularity schema $G_m = (g_m, g_{m-1}, \cdots, g_2, g_1)$. Let $\alpha = \langle (s_1, t_1) \rightarrow (s_2, t_2) \rightarrow \cdots \rightarrow (s_n, t_n) \rangle$ be the projection of s with respect to a multi-granularity sequence pattern prefix

$$\alpha = \alpha_1 \overset{[L_1, M_1, U_1]\mu_1}{\rightarrow} \alpha_2 \overset{[L_2, M_2, U_2]\mu_2}{\rightarrow} \cdots \overset{[L_{k-1}, M_{k-1}, U_{k-1}]\mu_{k-1}}{\rightarrow} \alpha_k, (k \leq n)$$

Temporal sequence $\gamma = \langle (s_k'', t_k) \rightarrow (s_{k+1}, t_{k+1}) \rightarrow \cdots \rightarrow (s_n, t_n) \rangle$ is called the suffix of s with respect to α, denoted as $\gamma = s/\alpha$, where $s_k'' = s_k - s_k'$.

Note, if α is not a multi-granularity sequence pattern prefix of s, the suffix of s with respect to α is empty.

This definition indicates that the suffix of a temporal sequence can be obtained by directly removing the prefix from its projection. In the example above, if the sequence $\alpha = \langle (c, 5) \rightarrow (abc, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$ is projected with respect to the β_2, then three different suffix are obtained as follows:

[Sid, b, 80]: $\langle (c, 80) \rightarrow (ab, 320) \rightarrow (d, 5400) \rightarrow (cf, 6400) \rangle$;

[Sid, b, 320]: $\langle (d, 5400) \rightarrow (cf, 6400) \rangle$.

Definition 11. Let α be a multi-granularity sequence pattern in a temporal sequence database D, The α-projected databases, denoted as $D|\alpha$, is the collection of suffixes of temporal sequences in D with respect to α.

Based on the above description, the newly proposed algorithm, MG-PrefixSpan, is shown as following:

Input: D: temporal sequence database; G_m: multi-granularity schema; min-sup: minimal support threshold.

Output: The complete set of frequent multi-granularity sequence patterns.

Method:

(01) scan D once, to find all frequent items, which are denoted by L^1.
(02) for each a L^1 do begin
(03) construct candidate temporal items $\alpha = (0, a)$ and add α in L_1;
(04) end
(05) output L_1;
(06) for each $s \in D$ do begin
(07) for each $\alpha \in L_1$ do begin
(08) for each β suffix of s with respect to α do begin
(09) if $\beta \neq null$ null then add β and its tag to $D|\alpha$;
(10) end
(11) if $D|\alpha \neq null$ null then call MG-PrefixSpan $(\alpha, 1, D|\alpha)$;
(12) end
(13) end

Subroutine MG-PrefixSpan $(\alpha, 1, D|\alpha)$

Parameter: α a multi-granularity sequence pattern; k the length of α; $D|\alpha$ the α-projected database.

(20) if $L^k = null$ then return;
(21) for all $a \in L^k$ and all $\mu \in G_m$ do begin
(22) construct candidate temporal item (A, a), where A=[L, M, U]μ or 0, add it in C_k;
(23) end
(24) for each s $s \in D|\alpha$ with tag [Sid, b, t] do begin
(25) for each $(A, a) \in C_k$ do begin
(26) for each (s_i, t_i) in s where a $a \in s_i$ do begin
(27) if $t_i == t$, then add 1to the counting number of (0, a) for different sid;

(28) if $g_j(1) \leq t_i - t < g_{j+1}(1)$ or $g_m(1) \leq t_i - t$ for $1 \leq j \leq m$, then do begin
(29) increase the counting number of (A, a) for different sid;
(30) let $L = min\{L, \lceil t_i - t \rceil^{g_j}\}$, $U = max\{U, \lceil t_i - t \rceil^{g_j}\}$, $M = M + \lceil t_i - t \rceil^{g_j}$;
(31) end
(32) end
(33) end
(34) end
(35) for each $(A, a) \in C_k$ do begin
(36) if counting number of (A, a) ≥ min-sup $|D|$, then do begin
(37) let M=M/the total number of pulsing M (A≠0);
(38) append (A, a) to α to form k+1-multi-granularity sequential pattern γ, add a in L^{k+1};
(39) let γ. support=min α. support, counting number of (A, a)/$|D|$, add γ in L^{k+1};
(40) end
(41) end
(42) output L^{k+1}
(43) for each $\gamma \in L_{k+1}$ do begin
(44) for each $s \in D|\alpha$ do begin
(45) for each suffix of s with respect to γ: β do begin
(46) if $\beta \neq null$ then add β and its tag to $D|\gamma$;
(47) end
(48) end
(49) if $D|\gamma = null$ then return;
(50) call MG-PrefixSpan $(\gamma, k + 1, D|\gamma)$;
(51) end
(52) return

The overall MG-PrefixSpan algorithm is summarized in the above. The most importance difference lies in that MG-PrefixSpan algorithm is able to deal with the temporal relationship between successive items in the patterns based on multi-granularities. Steps 21-34 handle the frequent items in $D|\alpha$ and correspond annotations with multi-granularities, where steps 21-23 construct candidate temporal items, steps 24-30 derive the low bound, upper bound and average time of each candidate temporal item from the projected database $D|\gamma$. If a candidate temporal item is frequent, then it can be appended to the prefix to form k+1-multi-granularity sequence pattern γ in the steps 35-41, where when a annotation of a item is 0, it only insert the item into the last element of the prefix, otherwise appends the temporal item to the prefix as an new element. Steps 43-48 construct a new, smaller projection $D|\gamma$, recursively find temporal items in $D|\gamma$ and yield longer multi-granularity sequence patterns until all the multi-granularity sequence patterns are found.

5 Experimental Results and Performance Analysis

In this section, we provide an experimental assessment of the proposed algorithm on synthetic data sets. The purpose is to test the performance of our algorithm MG-PrefixSpan. We will analyze the effects of input parameters on execution times and compare the performances with those of the most efficient algorithm PrefixSpan [6] and I-PrefixSpan [11]. The first algorithm being the fastest algorithm for sequential pattern mining without time intervals [6] and the second algorithm the most effective algorithm in finding sequential patterns with user-defined time intervals, It is natural to compare our algorithm with its.

Table 1: PARAMETERS USED IN THE GENERATION OF DATASET

|D|	Number of customers
|C|	Average number of transactions of per customer
|T|	Average number of items per transaction
|S|	Average length of maximal potentially large sequences
|I|	Average size of itemsets in maximal potentially large sequences
$\|N_S\|$	Number of maximal potentially large sequences
$\|N_I\|$	Number of maximal potentially large itemsets
|N|	Number if items
$\|T_T\|$	Average length of time intervals

Table 2: PARAMETER SETTINGS

Name	**|C|**	**|T|**	**|S|**	**|I|**	**|D|**	**Size (Mb)**
C10T2S4I1	10	2	4	1	10K	1.733
C10T2S4I2	10	2	4	2	10K	1.743
C10T2S8I1	10	2	8	1	10K	1.746
C10T4S4I1	10	4	4	1	10K	2.509
C20T2S4I1	20	2	4	1	10K	3.317

5.1 Evaluation Environment

The two algorithms are implemented by Sun Java language and experiments were conducted on a 1.7MHz Intel Pentium IBM laptop with 512MB main memory, running Microsoft Windows XP and Borland JBiulder 9.0 as the Java execution environment. Detailed algorithm implementation of PrefixSpan and I-PrefixSpan is according to the algorithms described in [4,9], but with the pseudo projection turned off; MG-PrefixSpan is implemented as described in this paper.

5.2 Synthetic dataset generation

In this work we extended the IBM synthetic generator described in [1,10] to generate synthetic data sets. Basically, each data-sequence is a list of transactions, where each transaction is a set of items, called itemset. However, the transaction data is extended so that the items in different itemsets are assigned different time values and that those in the same itemsets are assigned the same time values. The interval time value between successive itemsets for each customer obeys a Poisson distribution with mean w. the value w is drawn repetitively from a Poisson distribution with mean T_T for this particular customer [11]. After that, we determine the time t for each transaction of this customer by summing the interval times before the transaction.

Table 1 lists the parameters used in the generation of the simulating dataset. The parameters except for the last one are the classical ones used in the previous research. The parameter T_T is a new one used to generate the time for each transaction.

We generate datasets by setting $N_S = 500$, $N_I = 2500$, $N = 1000$ and $T_T = 60$. Table 2 summarizes the dataset parameter settings.

Table 3: PART OF THE EXACTED PATTE

Extracted patterns	**Support**(%)
$< (80) >$	22.89
$< (80) \overset{[1,3,6]day}{\rightarrow} (80) >$	1.71
$< (80) \overset{[1,2,5]day}{\rightarrow} (80) >$	1.97
$< (80) \overset{[1,3,6]day}{\rightarrow} (80) \overset{[1,2,5]day}{\rightarrow} (80) >$	0.54
$< (80) \overset{[2,3,6]day}{\rightarrow} (569) >$	0.62
$< (80) \overset{[1,2,4]day}{\rightarrow} (569) >$	0.84
$< (80, 432) >$	1.19
$< (80, 432) \overset{[1,2,3]day}{\rightarrow} (944) >$	0.60

Table 4: PART OF THE EXACTED PATTERN

Extracted patterns	**Support**(%)
$< (80) >$	22.89
$< (80) \rightarrow (80) >$	3.03
$< (80) \rightarrow (80) \rightarrow (80) >$	0.71
$< (80) \rightarrow (569) >$	1.39
$< (80, 432) >$	1.19
$< (80, 432) \rightarrow (944) >$	0.62

5.3 Time Granularity Selection

In the experiment, we use multi-granularity schema: G_3 = (Week, Day, Hour), and all the timestamps in the synthetic data sets are Hour, i.e. g_0 = Hour.

In order to compare the performance of algorithm I-PrefixSpan to that of MG-PrefixSpan, all the intervals between successive items are partitioned into five intervals: $\{l_0, l_1, l_2, l_3, l_4\}$, where $l_0 : t = 0 : 0 < t < 1$, $l_2 : 1 \leq t < 24$, $l_3 : 24 \leq t < 168$, and $l_4 : 168 \leq t < \infty$.

5.4 Comparison of Extracted Sequential Pattern Quality

The first test is a comparison of the quality of extracted patterns by algorithm PrefixSpan[6], I-PrefixSpan[11] and MG-PrefixSpan. The test is designed using the data set C10T4S5I2 and all minimum support thresholds is set to 0.005(0.5%). Table 3 shows part of the extracted frequent multi-granularity sequence patterns using MG-PrefixSpan algorithm, and Tables 4 and 5 show part of the extracted frequent sequential patterns using the algorithm PrefixSpan and I-PrefixSpan, respectively.

As shown in Table 3, we can made the following observation:

(1) once item 80 occurs, then item 80 will occur again within 1 day to 6 days, or with average time 3 days, with probability $(1.71/22.89) \times 100\% = 7.4\%$; within 1 week to 5 weeks, or with average time 2 weeks, with probability $(1.97/22.89) \times 100\% = 8.6\%$.

(2) once item 80 occurs and item 80 occurs within 1day to 6 days, or with average time 3 days, then item 80 will occur again within 1 week to 5 weeks, or with average time 2 weeks, with probability $(0.54/1.19)100\% = 45.4\%$.

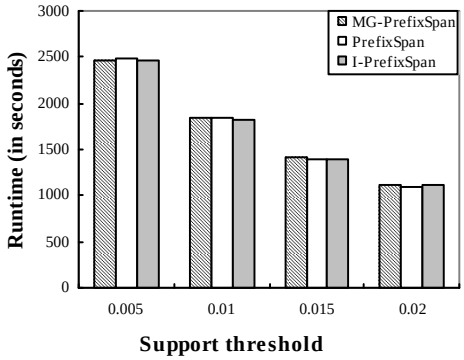

Figure 1: Performance of the of the algorithms on data set C10T2S4I1.

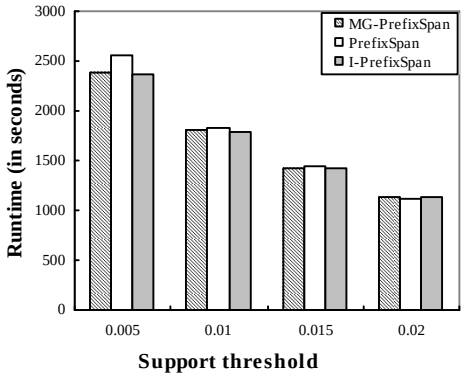

Figure 2: Performance of the algorithms on data set C10T2S4I2.

(3) once item 80 occurs, then item 569 will occur within 2 days to 6 days, or with average time 3 days, with probability $(0.62/22.89) \times 100\% = 2.7\%$; within 1 week to 4 weeks, or with average time 2 weeks, with probability $(0.84/22.89) \times 100\% = 3.7\%$.

(4) once item 80 and 432 occur, then item 94 will occur within 1 days to 3 days, or with average time 2 days, with probability $(0.60/1.19) \times 100\% = 50.4\%$;

As shown in Table 4, although the patterns are similar to those in Table 3, the time intervals between successive itemsets are not tighter than those in table 3. Thus users cannot precisely predict when items 569 and 944 occur and 80 occurs again. On the other hand, as shown in Table 5, there is no item annotation information based on multi-granularities in extracted sequences. Thus, users are not able to predict how long time item 569, 944 will occur and item 80 will occur again. Furthermore, the pattern $< (80) \rightarrow (569) >$ also makes users unable to distinguish periods of item 80 and 569 occurrence.

These results indicate that the multi-granularity sequence patterns mined by algorithm, MG-PrefixSpan, are more useful than those mined by algorithms, I-PrefixSpan, or PrefixSpan.

5.5 Comparison of the Execution Time

The second test of the three algorithms would compare the run times for different minimum supports. The comparison is on the five data sets shown in the table 2, where the minimum support threshold is varied from 0.5% to 2%. Fig.1 to Fig. 5 summarizes the results. It is clearly show that how the performance of the three algorithms changes as varying of the parameters $|\mathbf{C}|$, $|\mathbf{T}|$, $|\mathbf{S}|$ and $|\mathbf{I}|$, and the differences among the three algorithms.

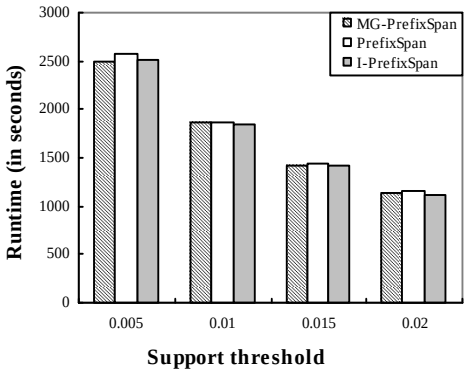

Figure 3: Performance of the algorithms on data set C10T2S8I1.

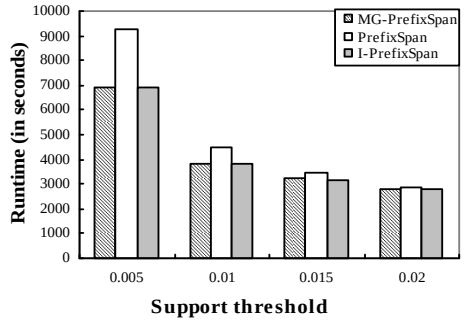

Figure 4: Performance of the algorithms on data set C10T4S4I1.

The results in the Fig. 1 Fig. 3 indicate that the processing time of algorithm MG-PrefixSpan is at different support threshold is no significant difference. The efficiency of MG-PrefixSpan is little less than that of the I-PrefixSpan. This result matches our expectation, because the MG-PrefixSpan algorithm does some more complicated computes for the bounds and average interval time in different granularities than that of the I-PrefixSpan.

On the other hand, the results in the Fig. 1, Fig. 2 and Fig. 3 show that the run time of three algorithms at different support threshold is also no significant difference. The speed of MG-PrefixSpan and I-PrefixSpan are little less than that of PrefixSpan. It is correct, because the algorithm PrefixSpan does not some complicated computes for interval time.

When we see the Fig. 3, Fig. 4 and Fig. 5 we can find that the run time of the three algorithms is increase rapidly as the minimum support threshold values vary from small to large. It is correct, because larger number of frequent sequential patterns could be found as the minimum support threshold value became smaller. However, it is worth note that the algorithm MG-PrefixSpan and I-PrefixSpan take less time than the PrefixSpan algorithm and the time difference of processing become larger as the minimum support threshold declines although the MG-PrefixSpan and I-PrefixSpan are more complicated than the PrefixSpan. In order to find the reason, Let us note the data sets. Fig. 3 test is performed on the data set C10T2S8I1, where the parameter average length of maximal potentially large sequences S increased from 4(Fig.1) to 8. On average, a potentially frequent sequential pattern consists of 8 transactions, which mean that although the average number of transactions in a potentially frequent sequential pattern is set to 8, the number of transactions in a sequence is still set to 10. This situation can not cause the number and length of frequent sequential pattern increase considerably, because the data set C10T2S8I1 (1.746MB) is as sparse as C10T2S4I1 (1.733MB); Fig.4 on the data set C10T4S4I1, where the parameter average number

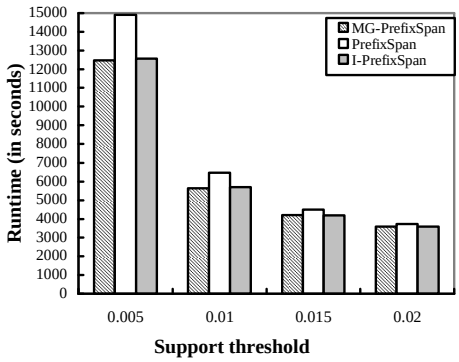

Figure 5: Performance of the algorithms on data set C20T2S4I1.

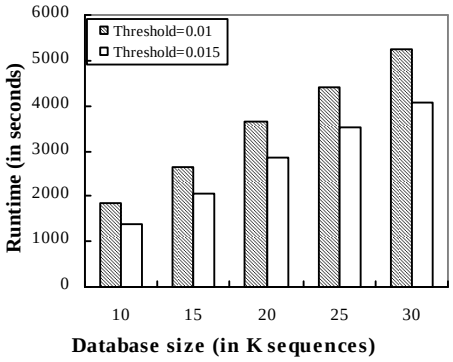

Figure 6: Scalability test of the algorithm MG-PrefixSpan on data set T2S4I1.

of items per transaction T increased from 2 (Fig.1) to 4, which mean, on average, a sequence consists of 10 transactions, each transaction composed of 4 items. It may result in the number of frequent sequential pattern increasing significantly; Fig.5 on the data set C20T2S4I1, where the parameter average number of transactions of per customer C increased from 10 (Fig.1) to 20, which mean, on average, a sequence consists of 20 transactions, each transaction composed of 2 items. It may result in the length of frequent sequential pattern increasing greatly. This both situations may cause the number and length of frequent sequential pattern increase considerably, because the data sets C10T4S4I1 and C20T2S4I1 are denser than the C10T2S8I1 and C10T2S4I1. When we use PrefixSpan to find all frequent sequential patterns, much more time would be spent in dealing with these more frequent sequential patterns although many of frequent sequential patterns may be useless. On the contrary, to the MG-PrefixSpan and I-PrefixSpan, although adding multiple time granularities or pseudo items to the traditional sequential pattern may make each of them to produce the several multiple granularities or time-interval patterns and need spend some time to does some complicated computation, many of them may not be frequent and some patterns found frequent by PrefixSpan may be infrequent by MG-PrefixSpan and I-PrefixSpan too. This reason reducing the numbers of frequent sequential patterns is why the MG-PrefixSpan and I-PrefixSpan are faster than the PrefixSpan.

5.6 Related global performances

The final test is the scalabilities of the MG-PrefixSpan algorithm. Four tests are designed using the data set C10T2S4I1 and all minimum support thresholds are set to 0.01 and 0.015. In each test, test how the runtime of the MG-PrefixSpan algorithm scales as one parameter is increased. Fig. 6 shows the result

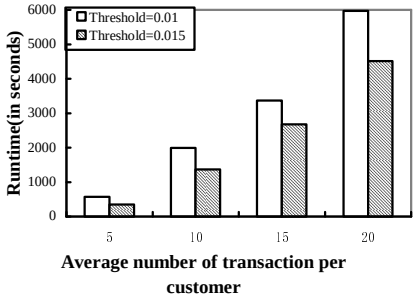

Figure 7: Performance of the algorithms on data set T2S10I1.

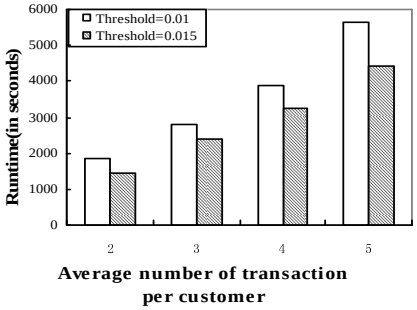

Figure 8: Performance of the algorithms on data set C10S10I1.

of scalability of the algorithm as the data set size grows from 10000 to 30000; Fig.7 shows the result as the average number of transactions of per customer varies from 5 to 20; Fig. 8 shows the result as the average number of items per transaction varies from 2 to 5. The results indicate that the MG-PrefixSpan algorithm has good scalabilities for its runtime increases linearly as each parameter varies from small to large respectively.

6 Conclusions

In this paper, we have proposed a novel approach for mining multi-granularities sequence patterns based on PrefixSpan [6], called MG-PrefixSpan. The multi-granularities sequence pattern not only reveals what items occur frequently together and in what order, but also the time interval that the next items occur after the preceding items. We use boundary interval and average time with multi-granularities, which are derived from the source data, rather than user-predetermined the time interval or only a typical time to annotate time interval between successive items in the patterns. The performance analysis shows that MG-PrefixSpan scales up linearly as the size of database, and has a good scalability with respect to length of sequence and the size of transaction.

The future work along this line of research includes several aspects as follows: (1) validation on large, real databases; (2) low-level optimizing mechanism of the algorithm.

Acknowledgment

This work was supported by the Department of Education of Shaanxi Province of China under Grant 05JK137 and the Natural Science Foundation of Shaanxi Province of China under Grant 2005F11.

Bibliography

[1] Constantinescu, Z, Marinoiu, C, Vladoiu, M, "Driving Style Analysis Using Data Mining Techniques," International Journal Of Computers Communications and Control, Vol.5, No.5, pp. 654-663, Dec 2010.

[2] Andonie, R, "Extreme Data Mining: Inference from Small Datasets," International Journal Of Computers Communications and Control, Vol.5, No.3, pp. 280-291, Sep 2010.

[3] R. Agrawal and R. Srikant, "Mining sequential patterns," Proc. of the 7th International Conference on Data Engineering (ICDE'95), pp. 3-14, March, 1995.

[4] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," Proc. of the 5th International Conference on Extending Database Technology, pp. 3-17, March, 1996.

[5] M. Zaki, "An Efficient Algorithm for Mining Frequent Sequences," Machine Learning, Vol. 40, pp. 31-60, 2000.

[6] J. Pei, J. Han and H. Pinto et al, "Mining Sequential Pattern-Growth: The PrefixSpan Approach," IEEE Transactions on Knowledge and Engineering, Vol.16, No.11, pp. 1424-1440, 2004.

[7] H. Manila, H. Toivonen, and A.I. Verkamo, "Discovery of frequent episodes in event sequences," Data Mining and Knowledge Discovery, Vol. 1, No. 3, pp. 256-289, 1997.

[8] M.N. Garofalakis, R. Rastogi and K. Shim, "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints," Proc. of the 25th International Conference on Very Large Data Bases (VLDB'99), pp. 223-234, September, 1999.

[9] J. Pei, J. Han and W. Wang, "Constraint-based Sequential Pattern Mining: The Pattern-growth Methods," Journal of Intelligent Information Systems, Volume 28, Issue 2, pp. 133-160, April, 2007.

[10] M. Yoshida et al. "Mining sequential patterns including time intervals", Proc. of SPIE Conf.-DMKD, pp. 213-220, April, 2000.

[11] Y.-L. Chen, M.-C. Chiang and M.-T. Ko, "Discovering time-interval sequential patterns in sequence databases," Expert System with Applications, Volume 25, Issue3, Pp. 343-354, October, 2003.

[12] R. Algawal and R. Srikant, "Fast algorithm for mining association rules in Large Databases.," Proc. of the 20th International Conference on Very Large Data bases (VLDB'94), pp. 487-499, September 1994.

[13] Y. Hirate, H. Yamana, "Generalized Pattern Mining with Item Intervals," Journal of Computers, Vol.1, No3, pp. 51-60, June, 2006.

[14] F. Giannotti, M. Nanni, and D. Pedreschi. "Efficient Mining of Temporally Annotated Sequences," Proc. of the 6th SIAM International Conference on Data Mining, pp. 346-357, April, 2006.

[15] C. Bettini, X.S. Wang and S. Jajodia et al, "Discovering Temporal Relationships with Multiple Granularities in Time Sequences," IEEE Transactions on Knowledge and Data Engineering, Vol. 10 (2), pp. 222-237, 1998.